Java Chapter 7: Object-Oriented Programming

Contents

Time required: 90 minutes

DRY

Don't **R**epeat **Y**ourself

Read: Think Java, 2nd Ed.

- Chapter 11 Designing Classes

Do: Online Tutorials

- Java OOP
- Java Classes/Objects
- Java Class Attributes
- Java Class Methods
- Java Constructors
- Java Encapsulation

Tutorial 7.1 - Sammy the Shark

The following program demonstrates classes, objects, and methods.

An object is an instance of a class. We'll construct a **Shark** object called **sammy**.

Create a Java class named: **Shark.java**

Create a Java program named: **SharkMaker.java**

```java
/**
 * Name: Shark.java
 * Written by:
 * Written on:
 * Purpose: Class to create a shark
 */

public class Shark {
   // Define class data attributes
   private String name;
   private int age;

   // Define class getters and setters
   public String getName() {
      return name;
   }

   public void setName(String name) {
      this.name = name;
   }

   public int getAge() {
      return age;
   }

   public void setAge(int age) {
      this.age = age;
   }

   // Define class methods
   public void swim() {
      // this references the class data attribute
      System.out.println(this.name + " is swimming.");
   }
}
```

```java
/**
 * Name: SharkMaker.java
 * Written by:
 * Written on:
 * Purpose: Demonstrate classes and objects
 */

public class SharkMaker {
   public static void main(String[] args) {
      // Create a Shark object named sammy
      System.out.println("Sammy the shark is created.");
      Shark sammy = new Shark();

      // Set the name and age
      sammy.setName("Sammy");
      sammy.setAge(3);

      // Call an object method
      sammy.swim();

      // Get the object data attributes
      System.out.println(sammy.getName() + " is " + sammy.getAge() + " years old.");
   }
}
```

Example run:

```
Sammy the shark is created.
Sammy is swimming.
Sammy is 3 years old.
```

Tutorial 7.2 - Constructing Sharks

A constructor is a special method is used to create an object from a class. It is run as soon as an object of a class is instantiated.

Classes are useful because they allow us to create many similar objects based on the same blueprint. This program creates two objects from the same class using the default constructor and a parameterized constructor.

1. Create a Java class named: **Shark2.java**
2. Create a Java program named: **SharkMaker2.java**
3. Copy the previous tutorial code into each file.

```java
/**
 * Name: Shark2.java
 * Written by:
 * Written on:
 * Purpose: Class with parameterized constructor to create a shark
 * If there is a parameterized constructor,
 * a default constructor must be put in manually
 */

public class Shark2 {
   // Define class data attributes
   private String name;
   private int age;

   // Default constructor
   public Shark2() {
   };

   // Constructor with two parameters
   public Shark2(String name, int age) {
      this.name = name;
      this.age = age;
   }

   // Define class getters and setters
   public String getName() {
      return name;
   }

   public void setName(String name) {
      this.name = name;
   }

   public int getAge() {
      return age;
   }

   public void setAge(int age) {
      this.age = age;
   }

   // Define class methods
   public void swim() {
      // this references the class data attribute
      System.out.println(this.name + " is swimming.");
   }
}
```

```java
/**
 * Name: SharkMaker2.java
 * Written by:
 * Written on:
 * Purpose: Demonstrate classes and objects
 */

public class SharkMaker2 {
   public static void main(String[] args) {
      // Create a Shark object named sammy
      // Use the default contructor
      System.out.println("\nSammy the shark uses the default constructor.");
      Shark2 sammy = new Shark2();

      // Set the name and age
      sammy.setName("Sammy");
      sammy.setAge(3);

      // Call an object method
      sammy.swim();

      // Get the object data attributes
      System.out.println(sammy.getName() + " is " +
            sammy.getAge() + " years old.");

      // Use the parameterized constructor
      System.out.println("\nSusie the shark uses the parameterized constructor.");
      Shark2 susie = new Shark2("Susie", 2);
      // Call the same methods as sammy
      susie.swim();

      // Get the object data attributes
      System.out.println(susie.getName() + " is " +
            susie.getAge() + " years old.");
   }
}
```

Example run:

```
Sammy the shark uses the default constructor.
Sammy is swimming.
Sammy is 3 years old.

Susie the shark uses the parameterized constructor.
Susie is swimming.
Susie is 2 years old.
```

Tutorial 7.3 - Coin Flip

The **Coin** class holds all the attributes and methods to flip a coin and return the value. The **CoinFlip** program creates a coin object and flips it 10 times.

Create a Java class named **Coin.java**

```java
/**
 * Name: Coin.java
 * Written by:
 * Written on:
 * Purpose: Class that flips a coin
 */

// Import library for random numbers
import java.util.concurrent.ThreadLocalRandom;

public class Coin {
    private final int MIN = 0;
    private final int MAX = 1;
    private int randomNumber;
    private String sideUp;

    public String Flip() {
        // Generate a random number between
        // MIN and MAX inclusive
        this.randomNumber = ThreadLocalRandom.current().nextInt(
                MIN,
                MAX + 1);

        // Determine which side was up
        if (randomNumber == 0) {
            this.sideUp = "Heads";
        } else {
            this.sideUp = "Tails";
        }
        // Return the side up
        return this.sideUp;
    }
}
```

Create a Java program that uses the Coin class named **CoinFlip.py** to flip a coin 10 times.

```java
/**
 * Name: CoinFlip.java
 * Written by:
 * Written on:
 * Purpose: Flip a coin object 10 times
 */

public class CoinFlip {
    Run | Debug
    public static void main(String[] args) {
        String tossResult = "";
        // Create a Coin object
        Coin quarter = new Coin();

        // Flip the coin 10 times
        for (int i = 0; i < 10; i++) {
            // Call object Flip Method, return a string
            tossResult = quarter.Flip();
            System.out.println(tossResult);
        }
    }
}
```

Example run:

```
Tails
Tails
Heads
Tails
Heads
Tails
Tails
Tails
Heads
Tails
```

Designing an Object-Oriented Program

The first step is to identify the classes the program will need.

1. Identify the real-world objects, the nouns.

   Customer
   Address
   Car
   labor charges
   Toyota

Some nouns contain other nouns. A Car can also be a Toyota. A customer would have an address. We need a Car and a Customer class.

2. Determine what describes or makes up the class, the data attributes.

   Customer: Name, Address, Phone

3. Determine the actions, the verbs, the methods.

   Move forward()
   Make purchase()
   create invoice()

   Customer: setName(), setAddress(), getAddress()

Keep everything that is relevant to the class in the class.

Assignment Submission

1. Attach the pseudocode.
2. Attach the program files.
3. Attach screenshots showing the successful operation of the program.
4. Submit in Blackboard.